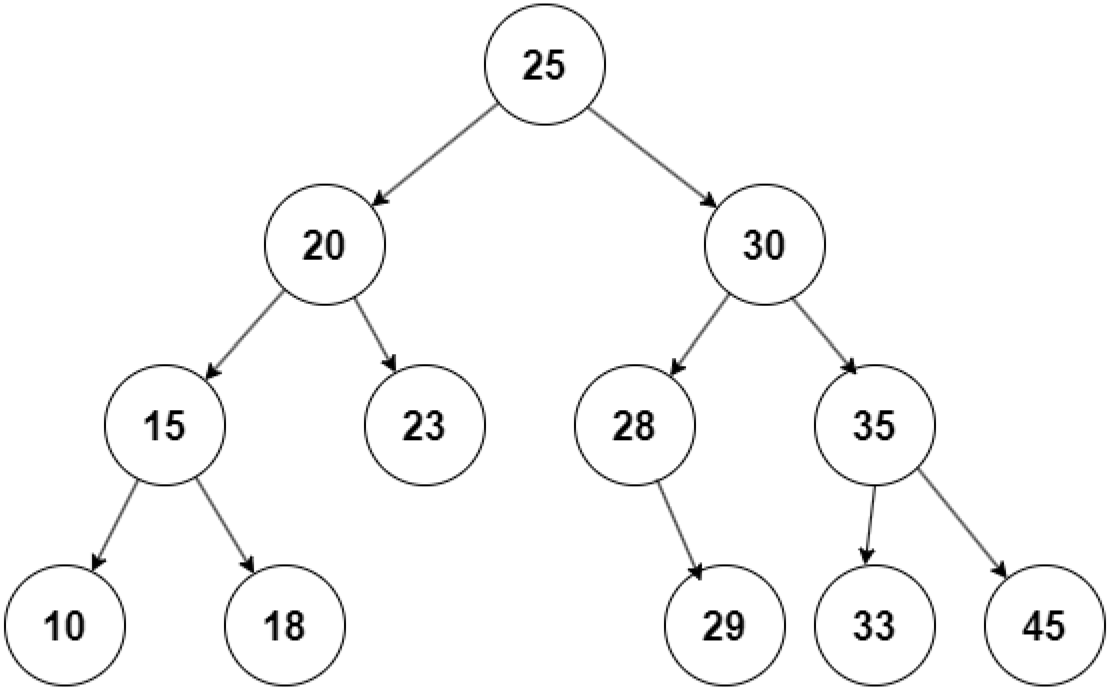
Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)

Question 1

Build a BST by inserting one element at a time. After each insert, please draw a picture. For this assignment, you can submit hand drawn pictures.

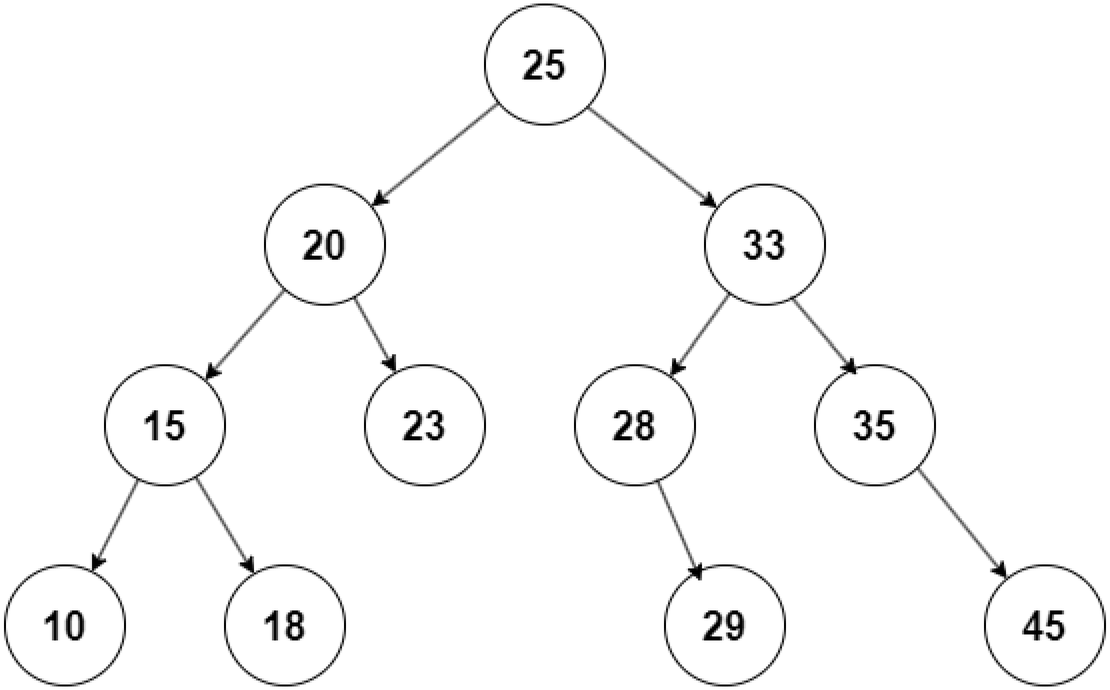
25, 20, 30, 28, 29, 15, 18, 23, 10, 35, 45, 33, 29



Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)

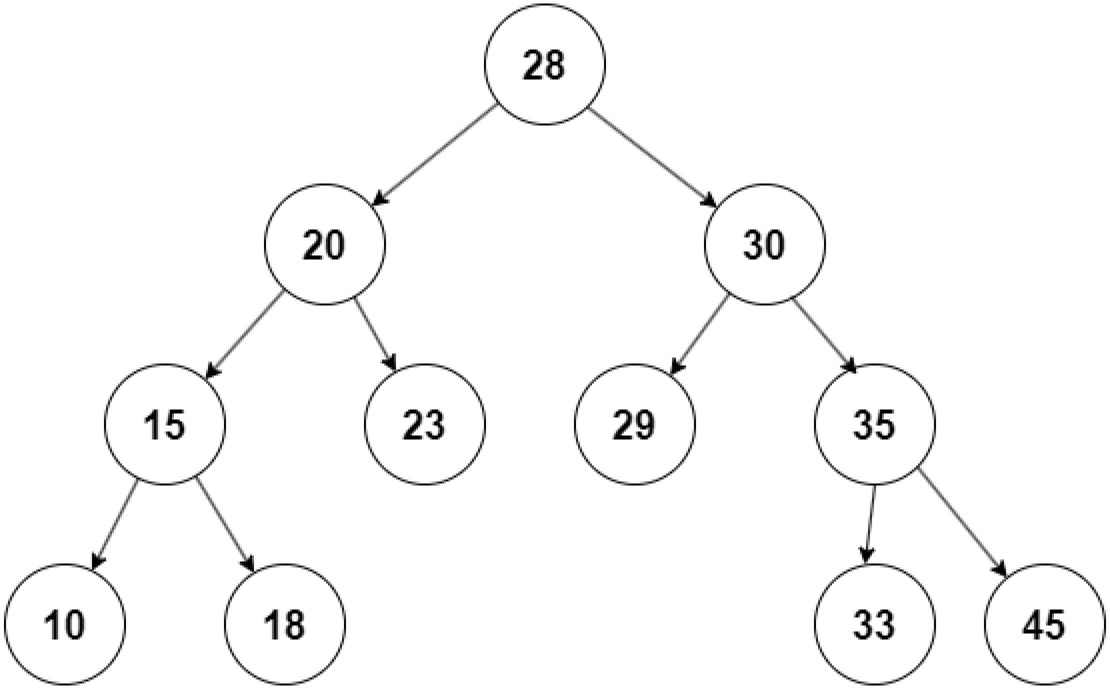
Question 2

A. Start from the BST you built. Delete 30



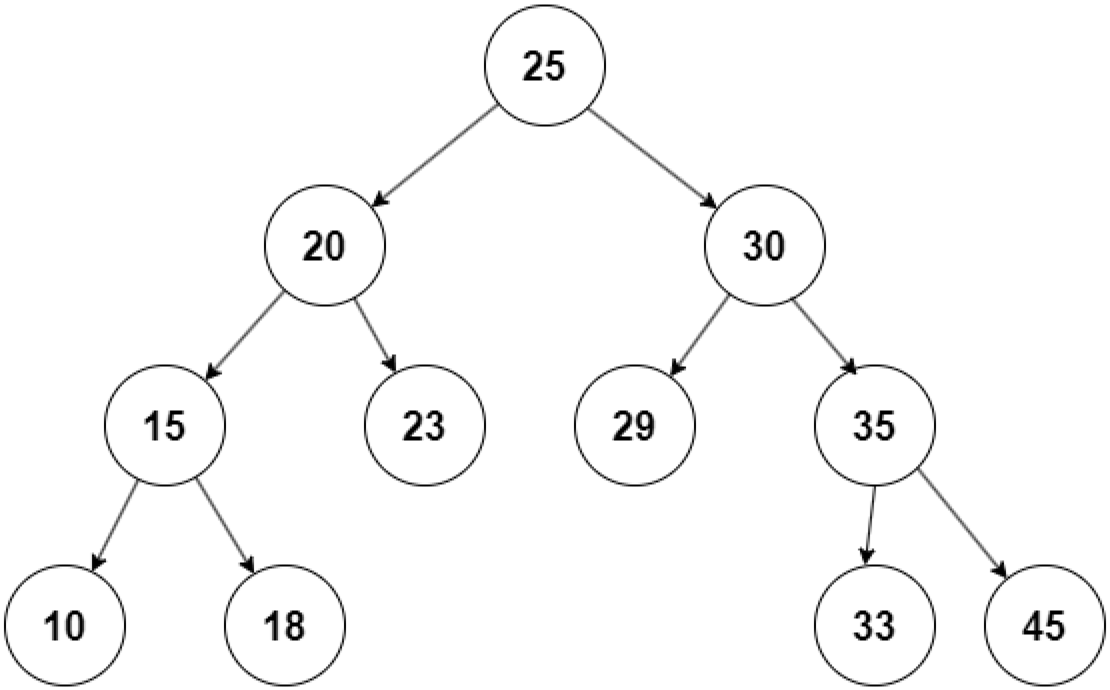
B. Start from the BST you built. Delete 25

Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)



C. Start from the BST you built. Delete 28

Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)



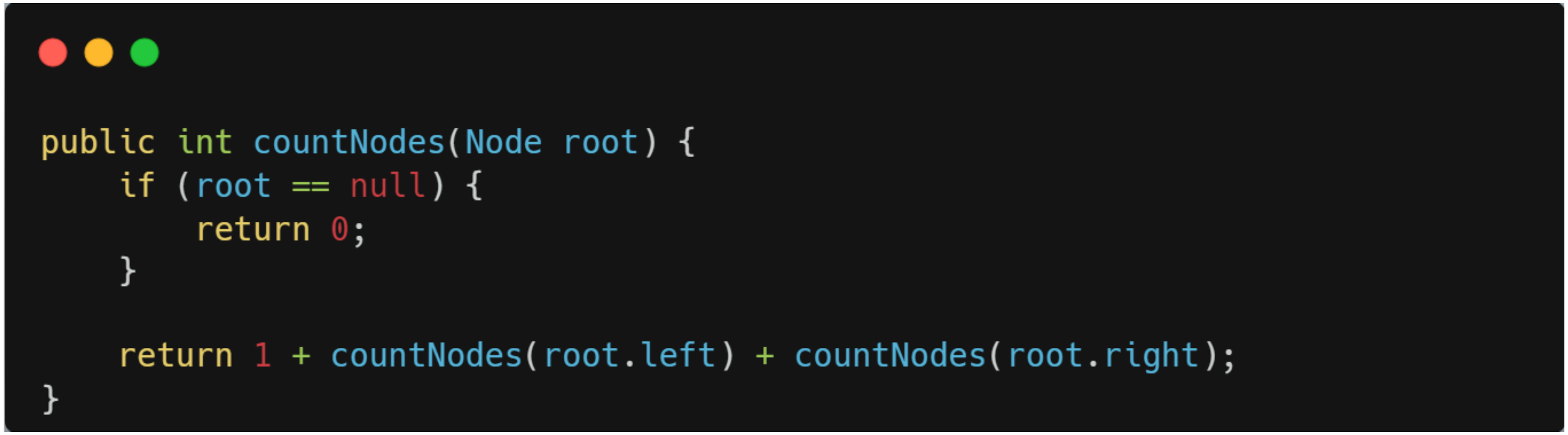
Question 3

A. Start from the BST you built. Preorder traversal
   => 25, 20, 15, 10, 18, 23, 30, 28, 29, 35, 33, 45
B. Start from the BST you built. Postorder traversal
   => 10, 18, 15, 23, 20, 29, 28, 33, 45, 35, 30, 25
C. Start from the BST you built. Inorder traversal
   => 10, 15, 18, 20, 23, 25, 29, 30, 33, 35, 45

Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)

Question 4. Write a recursive function to

A. count the nodes of a BST

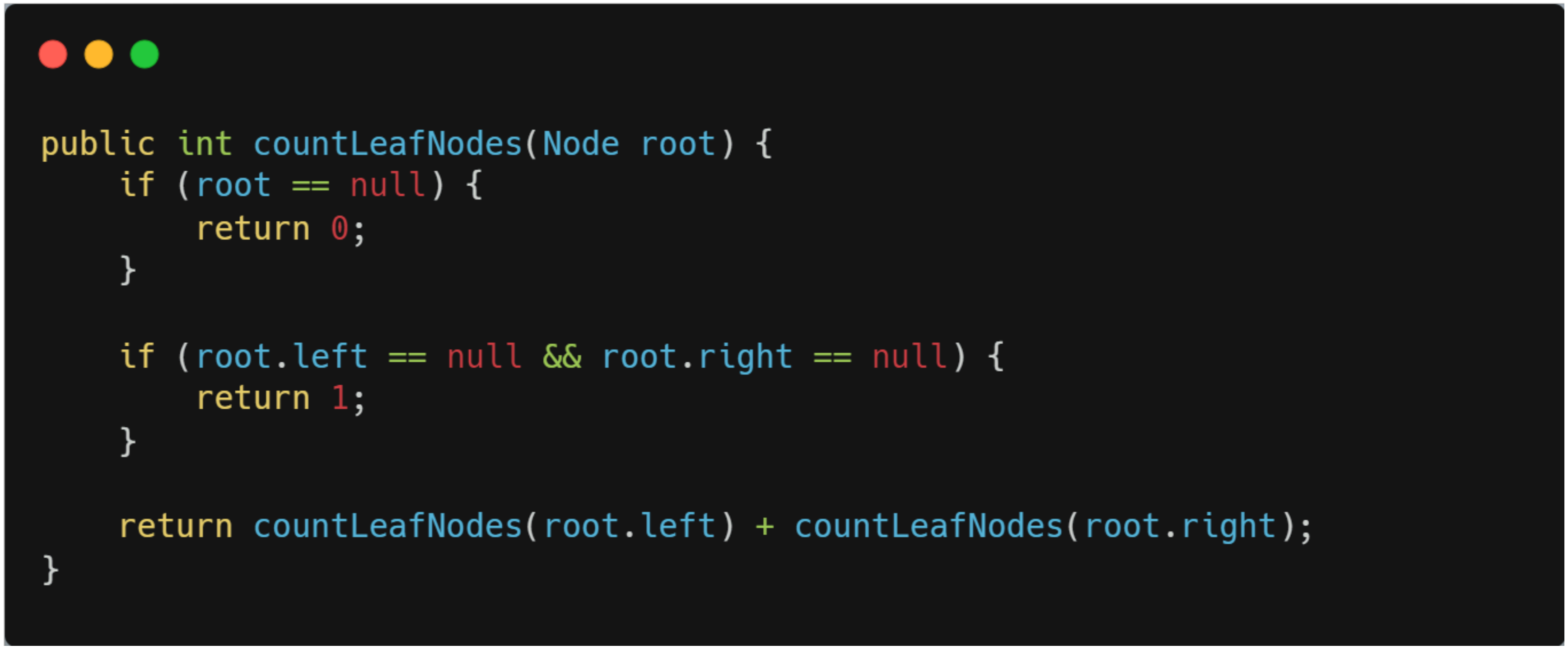


```java
public int countNodes(Node root) {
    if (root == null) {
        return 0;
    }

    return 1 + countNodes(root.left) + countNodes(root.right);
}
```

B. count the leaves of a BST

Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)



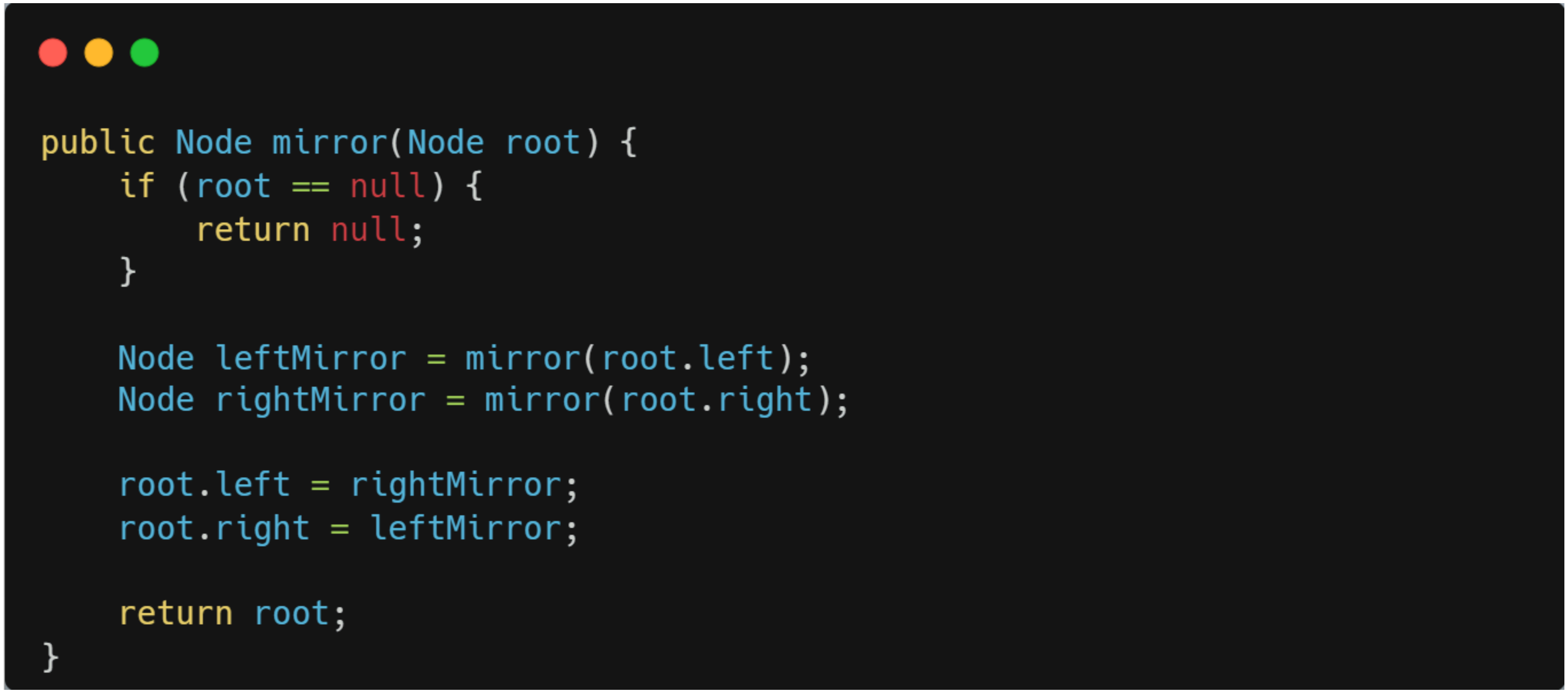
```
public int countLeafNodes(Node root) {
    if (root == null) {
        return 0;
    }

    if (root.left == null && root.right == null) {
        return 1;
    }

    return countLeafNodes(root.left) + countLeafNodes(root.right);
}
```

C. create a mirror image of the BST

Group members: Marwane Bidamou (4), Eduard Koshkelyan (8), Roshan Maharjan (11)



```java
public Node mirror(Node root) {
    if (root == null) {
        return null;
    }

    Node leftMirror = mirror(root.left);
    Node rightMirror = mirror(root.right);

    root.left = rightMirror;
    root.right = leftMirror;

    return root;
}
```